Ejercicios

Unidad 8

Array Y ArrayList

VECTORES

1. Crea un programa que pida diez números reales por teclado, los almacene en un array, y luego muestre todos sus valores.
2. Crea un programa que pida diez números reales por teclado, los almacene en un array, y luego muestre la suma de todos los valores.
3. Crea un programa que pida diez números reales por teclado, los almacene en un array, y luego lo recorra para averiguar el máximo y mínimo y mostrarlos por pantalla.
4. Crea un programa que pida veinte números enteros por teclado, los almacene en un array y luego muestre por separado la suma de todos los valores positivos y negativos.
5. Crea un programa que pida veinte números reales por teclado, los almacene en un array y luego lo recorra para calcular y mostrar la media: (suma de valores) / nº de valores.
6. Crea un programa que pida dos valores enteros N y M, luego cree un array de tamaño N, escriba M en todas sus posiciones y lo muestre por pantalla.
7. Crea un programa que pida dos valores enteros P y Q, luego cree un array que contenga todos los valores desde P hasta Q, y lo muestre por pantalla.
8. Crea un programa que cree un array con 100 números reales aleatorios entre 0.0 y 1.0, utilizando Math.random(), y luego le pida al usuario un valor real R. Por último, mostrará cuántos valores del array son igual o superiores a R.
9. Crea un programa que cree un array de enteros de tamaño 100 y lo rellene con valores enteros aleatorios entre 1 y 10 (utiliza 1 + Math.random()*10). Luego pedirá un valor N y mostrará en qué posiciones del array aparece N.
10. Crea un programa para realizar cálculos relacionados con la altura (en metros) de personas. Pedirá un valor N y luego almacenará en un array N alturas introducidas por teclado. Luego mostrará la altura media, máxima y mínima así como cuántas personas miden por encima y por debajo de la media.
11. Crea un programa que cree dos arrays de enteros de tamaño 100. Luego introducirá en el primer array todos los valores del 1 al 100. Por último, deberá copiar todos los valores del primer array al segundo array en orden inverso, y mostrar ambos por pantalla.
12. Crea un programa que cree un array de 10 enteros y luego muestre el siguiente menú con distintas opciones:

a. Mostrar valores.

b. Introducir valor.

c. Salir.

La opción 'a' mostrará todos los valores por pantalla. La opción 'b' pedirá un valor V y una posición P, luego escribirá V en la posición P del array. El menú se repetirá indefinidamente hasta que el usuario elija la opción 'c' que terminará el programa.

13. Crea un programa que permita al usuario almacenar una secuencia aritmética en un array y luego mostrarla. Una secuencia aritmética es una serie de números que comienza por un valor inicial V, y continúa con incrementos de I. Por ejemplo, con V=1 e I=2, la secuencia sería 1, 3, 5, 7, 9... Con V=7 e I=10, la secuencia sería 7, 17, 27, 37... El programa solicitará al usuario V, I además de N (nº de valores a crear).

14. Crea un programa que cree un array de enteros e introduzca la siguiente secuencia de valores: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, etc. hasta introducir 10 diez veces, y luego la muestre por pantalla.

~~**15.** Crea un programa que pida la usuario dos valores N y M y luego cree un array de tamaño N que contenga M en todas sus posiciones. Luego muestra el array por pantalla.~~

16. Crea un programa que cree un array de enteros e introduzca la siguiente secuencia de valores: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, etc. hasta introducir 10 diez veces, y luego la muestre por pantalla. En esta ocasión has de utilizar Arrays.fill().

17. Crea un programa que pida al usuario 20 valores enteros e introduzca los 10 primeros en un array y los 10 últimos en otro array. Por último, comparará ambos arrays y le dirá al usuario si son iguales o no.

18. Crea un programa que cree un array de tamaño 30 y lo rellene con valores aleatorios entre 0 y 9. Luego ordena los valores del array y los mostrará por pantalla.

19. Necesitamos crear un programa para mostrar el ranking de puntuaciones de un torneo de ajedrez con 8 jugadores. Se le pedirá al usuario que introduzca las puntuaciones de todos los jugadores (habitualmente valores entre 1000 y 2800, de tipo entero) y luego muestre las puntuaciones en orden descendente (de la más alta a la más baja).

20. Crea un programa que cree un array de tamaño 1000 y lo rellene con valores enteros aleatorios entre 0 y 99. Luego pedirá por teclado un valor N y se mostrará por pantalla si N existe en el array, además de cuantas veces.

MATRICES

21. Crea un programa que cree una matriz de tamaño 5x5 que almacene los números del 1 al 25 y luego muestre la matriz por pantalla.

22. Crea un programa que cree una matriz de 10x10 e introduzca los valores de las tablas de multiplicar del 1 al 10 (cada tabla en una fila). Luego mostrará la matriz por pantalla.

23. Crea un programa que cree una matriz de tamaño NxM (tamaño introducido por teclado) e introduzca en ella NxM valores (también introducidos por teclado). Luego deberá recorrer la matriz y al final mostrar por pantalla cuántos valores son mayores que cero, cuántos son menores que cero y cuántos son igual a cero.

24. Necesitamos crear un programa para almacenar las notas de 4 alumnos (llamados "Alumno 1", "Alumno 2", etc.) y 5 asignaturas. El usuario introducirá las notas por teclado y luego el programa mostrará la nota mínima, máxima y media de cada alumno.

25. Necesitamos crear un programa para registrar sueldos de hombres y mujeres de una empresa y detectar si existe brecha salarial entre ambos. El programa pedirá por teclado la información de N personas distintas (valor también introducido por teclado). Para cada persona, pedirá su género (0 para varón y 1 para mujer) y su sueldo. Esta información debe guardarse en una única matriz. Luego se mostrará por pantalla el sueldo medio de cada género.

ArrayList

Ejercicio 1 – Alturas

Realiza un programa que tenga un ArrayList llamado AlturaAlumnos que mantenga una lista con las alturas de los alumnos de un centro. Serán valores positivos entre 0,50 y 2,50 redondeados a dos decimales. El programa tendrá las siguientes funciones (accesibles mediante un menú):

a) Añadir altura.

b) Mostrar lista actual con el número de posición.

c) Eliminar por posición. Se le pasa como parámetro una posición y elimina la altura en dicha posición.

d) Eliminar por valor. Se le pasa como parámetro una altura y elimina todas las posiciones en las que se encuentre dicha altura. Devuelve la cantidad de

eliminaciones.

e) Ordenar la lista

Ejercicio 2- Divisores

Realizar un programa que tenga una función a la que se le pasa un entero y devuelva un ArrayList con todos sus divisores.

Ejercicio 3 - Productos

Supongamos una clase Producto con dos atributos:

- String nombre
- int cantidad

Implementa esta clase con un constructor (con parámetros) además de los getters y setters de sus dos atributos. No es necesario comprobar los datos introducidos.

A continuación, en el programa principal haz lo siguiente:

1. Crea 5 instancias de la Clase Producto.
2. Crea un ArrayList.
3. Añade las 5 instancias de Producto al ArrayList.
4. Visualiza el contenido de ArrayList utilizando Iterator.
5. Elimina dos elemento del ArrayList.
6. Inserta un nuevo objeto producto en medio de la lista.
7. Visualiza de nuevo el contenido de ArrayList utilizando Iterator.
8. Elimina todos los valores del ArrayList.

Ejercicio 4 – Factura

Diseñar una clase Factura que consta de:

- Número identificador: lo proporciona el usuario en el alta de la factura.
- Fecha de la factura: la toma del sistema en el momento del alta.
- Número de cliente: : lo proporciona el usuario en el alta de la factura.
- Porcentaje de IVA: 21% para todas las facturas.
- Un número indeterminado de lineaFactura que contienen:

    - Descripción del producto
    - Precio unitario
    - Cantidad de unidades
    - Porcentaje de descuento: 5% para líneas con más de 10 unidades.
    - Importe total de la línea.
- Importe total: inicialmente cero, y se va actualizando siempre que se añadan/eliminen líneas.

Implementar la clase con su constructor y métodos para añadir línea de factura, eliminar línea de factura y mostrar la factura por consola. Te hará falta una clase LineaFactura con su constructor. Para añadir línea de factura, habrá que solicitar al usuario los campos necesarios para añadirlo (descripción, precio unitario y cantidad de unidades). Para eliminar una línea, solicitaremos el número de línea.

Hacer también un programa con un menú para gestionar una factura (alta, añadir/eliminar líneas, mostrar factura) Nota: pensar en método toString() para LíneaFactura.

Ejercicio 5 – Distribución

Realizar un programa que cree un ArrayList con 10.000 números aleatorios entre 1 y 6 (como si fuese lanzar un dado). Utilizando los métodos estáticos de la clase Collections guarda en otro ArrayList la distribución de resultados obtenidos (cuantas veces ha salido el uno, cuantas veces ha salido el dos, etc...) y muestra su contenido. Finalmente, también mediante métodos de Collections, mostrar la diferencia de veces entre el número que más ha salido y el que menos ha salido.

Tercera parte

Ejercicio 6 - Receta

Implementa las clases Ingrediente y Receta (partiendo de los modelos UML que están anexos abajo). Una vez creadas las clases. Implementa un método main que introduzca por teclado los valores de los atributos para crear una receta. Se preguntará cuántos ingredientes lleva la receta. Una vez introducidos los valores mostrar la receta.

Receta
-Nombre(String) -Elaboracion(String) -Duración(int) -ingredientes (Ingrediente [])
+Receta(String, String, int, Ingrediente[]) +setNombre(String):void +getNombre():String +setElaboracion(String):void +getElaboracion():String +setDuracion(int):void +getDuracion():int +mostrarReceta():void

Ingrediente
-Nombre(String) -Cantidad (int) -Unidad(String)
+Ingrediente(String, float, String,) +setNombre(String):void +getNombre():String +setCantidad(int):void +getCantidad():int +setUnidad(String):void +getUnidad():String

Ejercicio 7 - EticalBank

Usando el ejercicio del tema anterior, modifícalo para que contemple las siguientes especificaciones.

La clase CuentaBancaria tendrá un nuevo atributo, movimientos, que será un ArrayList para guardar todos los movimientos que se realizan en la cuenta.

Cuando se ingrese o se retire dinero de la cuenta se verá reflejado en la lista de movimientos de la cuenta.

Añadiremos una nueva opción al menú de nuestra aplicación, “Mostrar movimientos” que mostrará todos los movimientos que se han realizado en la cuenta.

Ejercicio 8 – La tienda del barrio

En La Tienda del Barrio S.A. necesitan un programa en lenguaje Java que les ayude a gestionar la información sobre los artículos que tienen en la tienda, así como realizar ventas a clientes y compras a proveedores.

Clase 'Articulo'

- Un artículo tiene un identificador (número entero), un nombre, un precio de venta a cliente, un precio de compra a proveedor, un IVA (%) y un stock (representa la cantidad de ese artículo disponible en tienda). El nombre y el identificador de un artículo no pueden cambiar. El identificador ha de establecerse automáticamente y ser único, podemos llevar un contador de los objetos creados y así el identificador será único. El IVA es el mismo para todos los artículos (un 21%). Tanto el precio de compra como el precio de venta han de ser valores superiores a cero. El precio de venta ha de ser superior al precio de compra. El stock ha de ser superior o igual a cero.
- Deberá tener un único constructor así como todos los getters y setters que sea posible.
- Deberá tener al menos dos métodos públicos, uno para vender (a cliente) y otro para comprar (a proveedor). En ambos casos deberá pasarse un único argumento con la cantidad de unidades a vender o comprar, y se devolverá un valor indicando si fue posible realizar la operación o no. Si la operación se puede realizar, se deberá modificar el stock del artículo.
- Deberá tener un método público que devuelva una cadena de texto de una sola línea con la información sobre dicho artículo: id, nombre, precios, IVA y stock.
- En los casos en los que sea necesario, los métodos públicos mostrarán un mensaje por System.err si no es posible realizar la operación solicitada.
- Pueden implementarse otros métodos si se considera necesario.
- Es obligatorio que esta clase no realice ningún tipo de entrada por teclado ni salida por pantalla (Excepto los mencionados mensajes de error).

Clase 'Tienda'

- Contendrá la función 'main' del programa.
- Al iniciar el programa se mostrará por pantalla un menú principal con las siguientes opciones:

1. Mostrar artículos.
2. Venta a cliente.
3. Compra a proveedor.
4. Gestionar artículos.
5. Salir.

• La opción 1 mostrará por pantalla la descripción de todos los artículos de la tienda.

• La opción 2 permitirá realizar una venta. Pedirá los identificadores y cantidades de los artículos deseados además del nombre del cliente. Mostrará el precio total y pedirá confirmar la venta.

• La opción 3 permitirá realizar una compra. Pedirá los identificadores y cantidades de los artículos deseados además del nombre del proveedor. Mostrará el precio total y pedirá confirmar la compra.

• La opción 4 mostrará un submenú con cuatro opciones: '1. Añadir artículo', '2. Editar artículo', '3. Eliminar artículo' y '4. Volver'. Las tres primeras opciones pedirán introducir la información necesaria y realizarán la operación si es posible. La cuarta opción volverá al menú principal.

• La opción 5 termina el programa.

•El menú principal y el submenú se volverán a mostrar tras cada operación hasta que el usuario elija 'Salir' o 'Volver' según el caso.

• Toda interacción con el usuario deberá realizarse por entrada y salida estándar (teclado y pantalla).

• Los artículos deberán almacenarse en memoria en alguna estructura de datos.

• Deberán manejarse los posibles errores y que puedan producirse.

• No programes todo directamente en la función main. Implementa distintas funciones adicionales para que el código sea lo más modular.

• Es obligatorio utilizar la clase 'Articulo'.

Ejercicio 9 - Jugando al...

Implementa el método lineas que recibe como parámetro una matriz de números enteros que representa el tablero de un juego (de cualquier número de filas y columnas) y retorna como resultado un ArrayList con las filas donde se haya producido una “línea”. Cada elemento de la matriz del tablero tendrá un número entero que representa si es está ocupada por un bloque o no. Cada número entero podrá ser:

0 – casilla vacía

1 – casilla ocupada por un bloque de color rojo

2 – casilla ocupada por un bloque de color verde

3 – casilla ocupada por un bloque de color azul

En el juego, se considera que se ha conseguido una “línea” cuando todas las casillas de la misma fila de la matriz están ocupadas por bloques.

Por ejemplo, dada la matriz A:

0	0	0	0	0
0	0	0	0	0
0	0	0	0	3
0	2	2	0	3
2	2	1	1	3
0	3	2	3	3
1	3	2	2	1
3	3	1	2	1

Líneas serían las filas 4, 5 y 7, por lo que método retornará este ArrayList

4 6 7

Para tus pruebas puedes usar esta matriz

int[][]

tablero={{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,3},{0,2,2,0,3},{2,2,1,1,3},{0,3,2,2,1},{1,3,2,2,1},{3,3,1,2,1}};

Ejercicio 10

Implementa un programa que cree una matriz de tamaño F x C (estos valores se introducirán como parámetros en la línea de comandos) y la rellene con números enteros aleatorios entre 1 y 20. Luego pedirá por teclado un número V y mostrará por pantalla cuántos valores de cada fila son múltiplos de V. Supondremos que el usuario introducirá valores válidos por lo que no será necesario comprobarlos.

Por ejemplo, dada esta matriz de 3x3 y V=5

5	10	9
1	6	19
20	15	10

El programa mostrará por pantalla:

Fila 1: 2

Fila 2: 0

Fila 3: 3

Ejercicio 11

Realiza un programa que construya una matriz de 3x3 y la rellene aleatoriamente de números 0 o 1. El programa mostrará el tablero que se ha generado y cuantas '3 en raya' tienen los 0's y cuantas tienen los 1's, indicando quien en el ganador y si han empatado. Hay que tener en cuenta las filas, las columnas y las diagonales principal y secundaria.

Por ejemplo, dada esta matriz, mostrará Lineas 0: 2 Lineas 1: 0 Gana 0

0	1	1
0	1	0
0	0	0